A Practical Approach for Performance Analysis of Shared-Memory Programs

Bogdan Marius Tudor and Yong Meng Teo
Department of Computer Science
National University of Singapore

18 May 2011
25th IEEE International Parallel & Distributed Processing Symposium, Anchorage, USA

Outline

- Motivation
- Objective
- Model Overview
  - Data Dependency
  - Memory Contention
  - Limitations
- Evaluation
- Related Work
- Conclusions

Motivation

- Understanding impact of parallel programming choices
  - Many languages, models and methodologies for shared-memory programs
- Trade-offs in existing approaches

	Intrusiveness	Accuracy	Difficult to Apply
Analytical Models	No	No	No
Instrumentation and Trace-driven Analysis	Yes	Yes	Yes
Empirical Approaches	No	Yes	Yes

Objective

A practical model for estimating speedup and speedup loss due to data dependency and memory contention in shared-memory programs.

Practical:

- Independent of language and threading implementation
- No instrumentation of source or binary code

Contributions

1. Analytical model for speedup and speedup loss for shared-memory programs on multi-socket UMA and NUMA machines

2. Analysis of speedup and speedup loss for six dwarfs from NPB 3.3 benchmark

Outline

- Motivation
- Objective
- Model Overview
  - Data Dependency
  - Memory Contention
  - Limitations
- Evaluation
- Related Work
- Conclusions

Model Overview

Shared Memory Program: *m* threads, *n* cores

Useful Work	Memory Contention	Data Dependency

Number of Threads: m

Active Threads: $A(m,n)$

Data dep: $m - A(m,\infty)$

Useful Work: $U(n)$

Contention: $M(n)$

$$\omega(n) = \frac{M(n)}{U(n)}$$

$$\boxed{S(m,n) = \frac{A(m,n)}{1+\omega(n)}}$$

Data Dependency Model

- Objective: derive $A(m,n)$
- Run the program with *m* threads on *b* cores $m > b$, which leads to threads queueing

$$A(m,\infty,t) = x(m,t) + q(m,t)$$

#active threads,
given unbounded #cores

#running threads

#threads queuing for
service in OS run-queue

- Determine $A(m,n)$ as time weighted average from trace of run-queue

Memory Contention Model

- Objective: $\omega(n)$
- Focus is memory contention among cores

$$C(n) = W + B + M(n)$$

total #cycles across n cores

#work cycles

#stall cycles unrelated to contention

#stall cycles due to contention

$$\omega(n) = \frac{M(n)}{W+B} = \frac{C(n)-C(1)}{C(1)} \qquad C(n) = ?$$

Memory Contention Model

- M/M/1 model for $C(n)$
- Single-socket: $$C(n) = \frac{r(n)}{\mu - n\lambda}$$
- Multi-socket
  - UMA: $$C(n) = C(c) + C(n-c) + \Delta C$$
  - NUMA: $$C(n) = C(c) + r(n)\delta(n-c)$$
- Linear regression for λ, μ, δ and ΔC
  - Two values of $C(n)$ for single-socket
  - Three values of $C(n)$ for multi-socket

Model Summary

$$S(n) = \frac{A(m,n)}{1+\omega(n)}$$

Derived from a trace of the OS run-queue on a single baseline run

Derived from HW counters measurement using two or three runs

- Practical
  - No instrumentation
  - Independent of programming language
  - Independent of threading package
  - Requires ≤3 runs

Limitations

- Data-dependency model
  - Programs with programming language tasks
  - Programs with busy-waiting synchronization
  - Programs with significant I/O
- Memory contention model
  - Programs with very low memory contention
  - Heterogeneous memory affinity among threads
  - Heterogeneous cores or memory nodes

Outline

- Motivation
- Objective
- Model Overview
  - Data Dependency
  - Memory Contention
  - Limitations
- Evaluation
- Related Work
- Conclusions

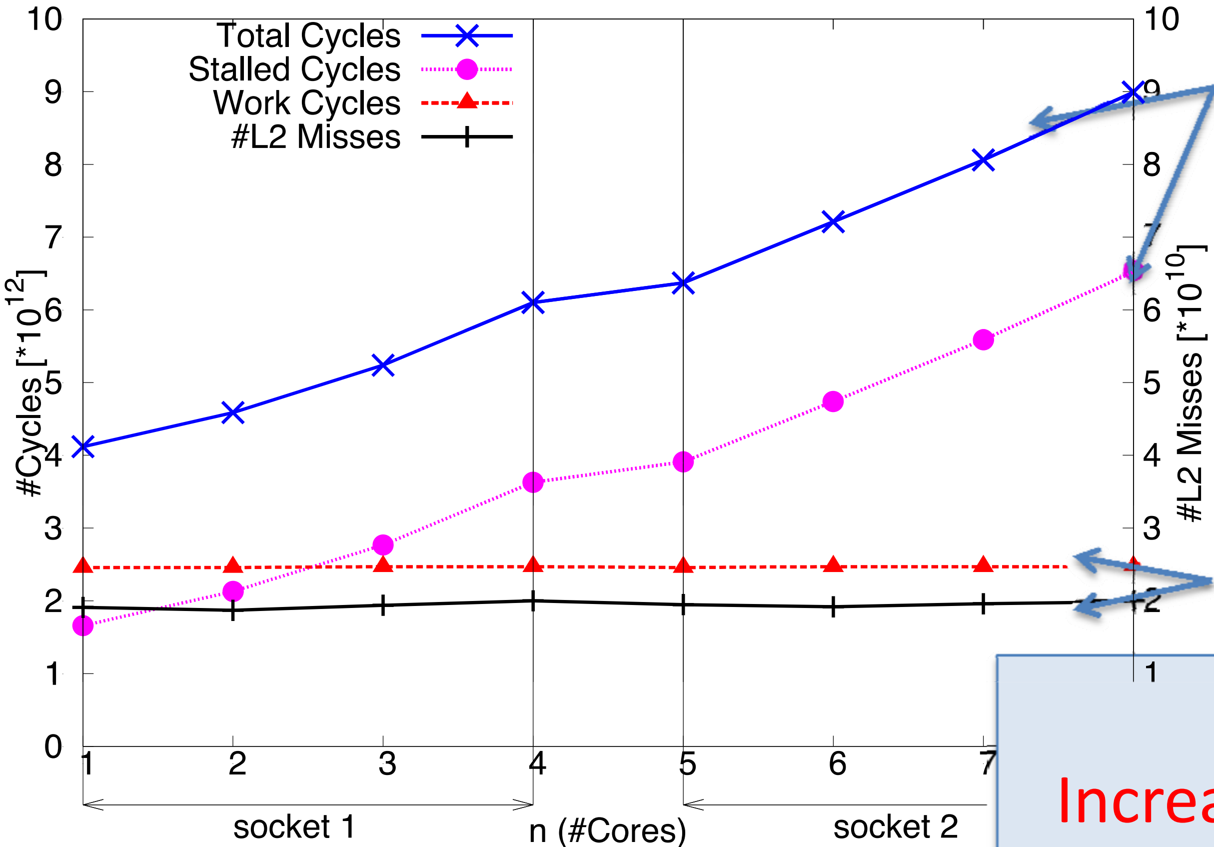
Evaluation Setup

- Workload: NPB 3.3

Name	Description of Parallel Kernel
EP	Embarrassingly parallel: low data dependency
IS	Parallel sorting: bucket sort on Integers
FT	Spectral methods: fast Fourier transform
BT	Dense linear algebra: use matrix to store data
CG	Sparse linear algebra: matrix with many 0 values
SP	Structured grid: pentadiagonal solver

  – GCC 4.2 with full optimizations
- Systems:
  – UMA: 8 cores Intel E5320, 1.87 GHz, 4 GB DDR2
  – NUMA: 16 cores Intel E5520, 2.27 GHz, 24 GB DDR3

Evaluation



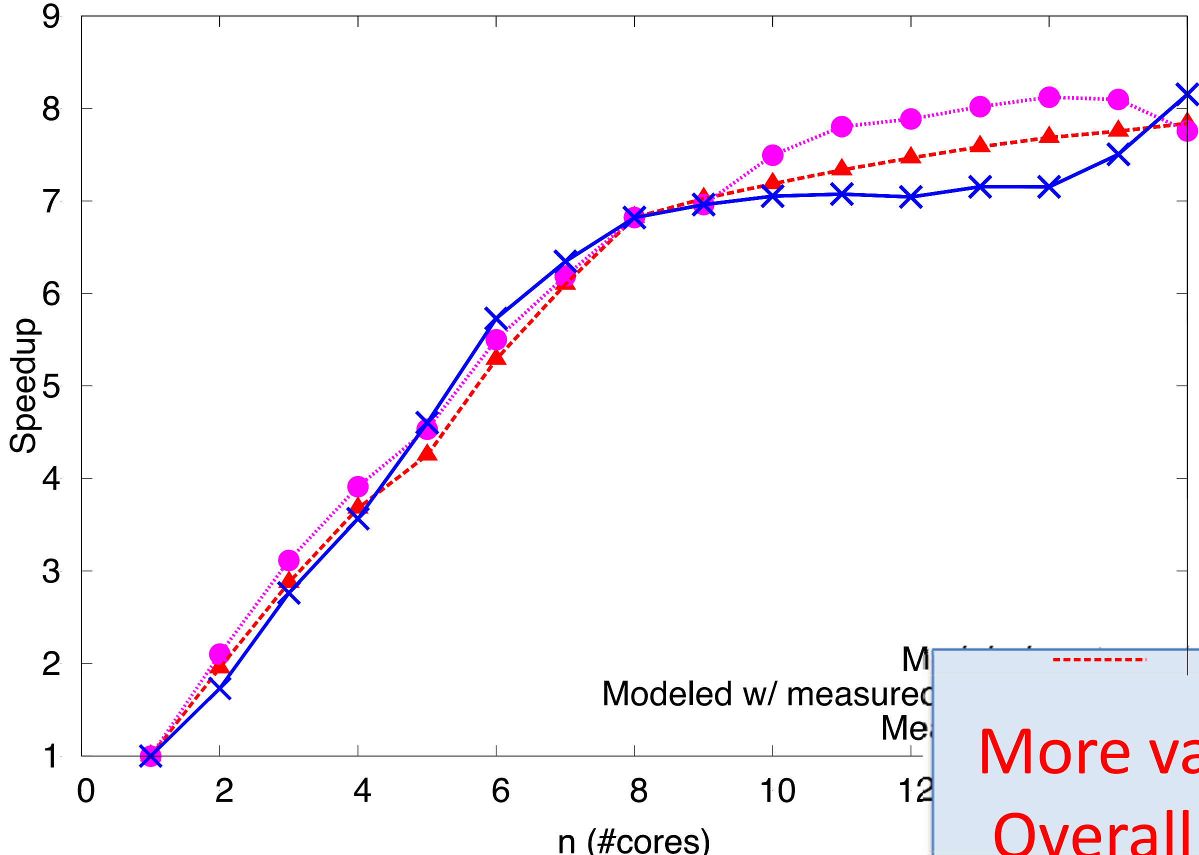
BT.C on UMA, m=8 threads

The growth in the #cycles is correlated with the stall cycles

The ratio of work cycles to last level cache misses is constant

Increase in #cycles is caused by contention among cores

Validation



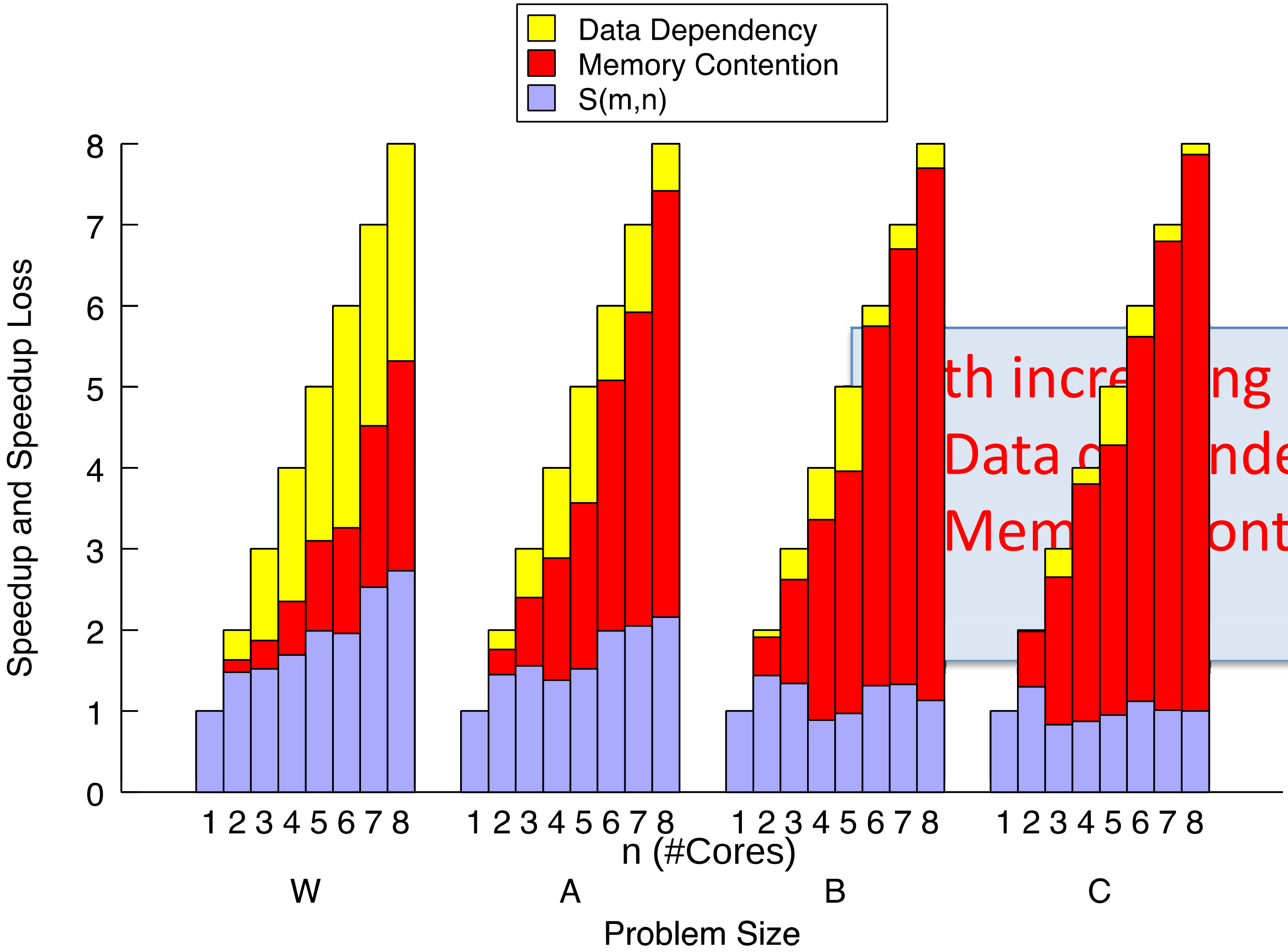
BT.C on NUMA, m=16 threads

A(m,n): baseline run on 1 core
ω(n): runs on 1, 8 and 9 cores

More validation results in paper. Overall average relative error is 9% for UMA and 11% for NUMA

Effect of Problem Size

SP, m=8 threads



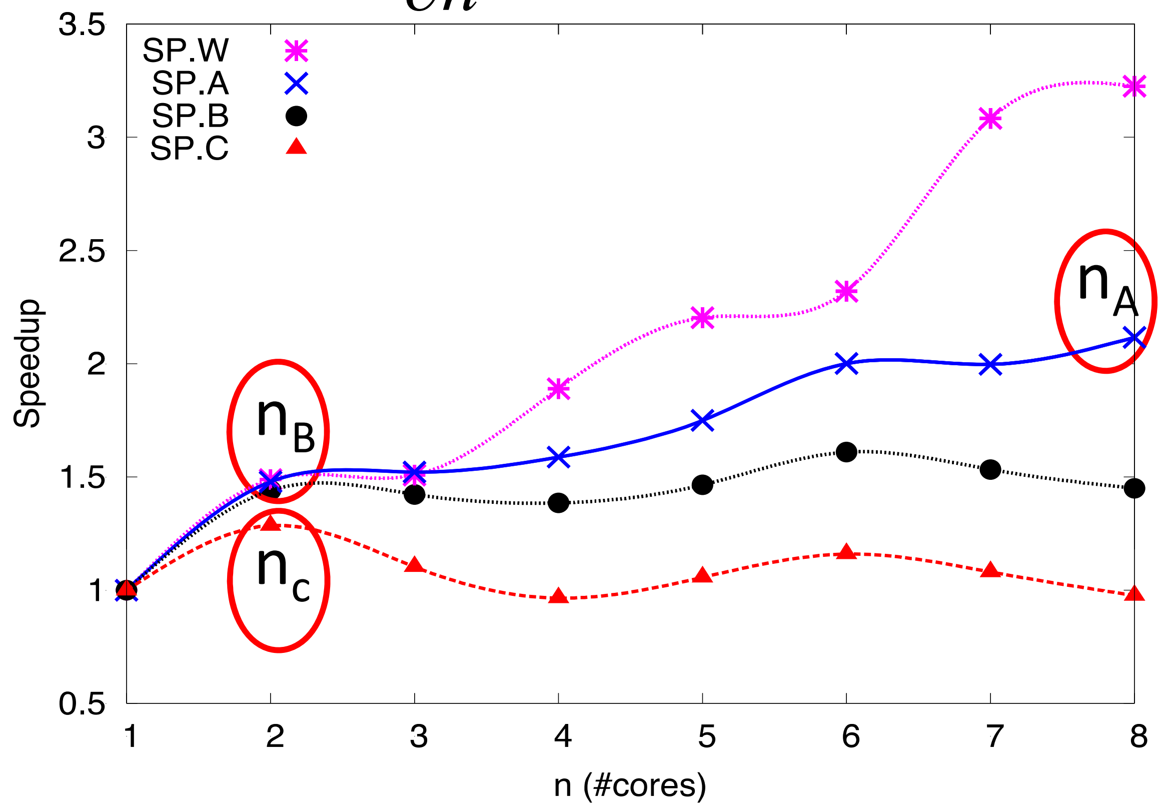
th incre ng problem size:
Data d ndency decreases
Mem ontention increases

Predicting Optimal #Cores

- Optimum n is $max\{S(m,n)\}$ from the range

$$\frac{d\omega(n)}{dn} < S(m,n)\frac{\partial A(m,n)}{\partial n}$$

- SP on UMA
  - $n_W = 12$
  - $n_A = 8$
  - $n_B = 2$
  - $n_C = 2$



Outline

- Motivation
- Objective
- Model Overview
  - Data Dependency
  - Memory Contention
  - Limitations
- Evaluation
- Related Work
- Conclusions

Related Work

- Instrumentation: OPARI, PMPI, PIN
- Empirical methods:
  - Regression: Barnes et al. (2008), Curtis-Maury et al. (2008)
  - Machine learning: Ghanapathi et al. (2009)
  - Neural networks: Singh et. al (2010)
- Analytical models:
  - Amdahl's law (1967)
  - DAG: Eager et al. (1989), Downey (1997)
- Run-queue size used for capacity planning of web-servers: Kelly et. al (2008)

Conclusions

- Analytical model for speedup and speedup loss due to data dependency and memory contention in shared-memory programs
  - Inputs derived from at most 3 runs
  - UMA and NUMA systems
  - Practical
- Measurement validation of 9%, 11% error
- Application of the model is determining the number of cores that optimizes speedup

Q&A

Thank you!

[teoym,bogdanma]@comp.nus.edu.sg

B. Tudor and Y.M. Teo, **A Practical Approach for Performance Analysis of Shared Memory Programs**, Proceedings of 25th IEEE International Parallel & Distributed Processing Symposium, IEEE Computer Society Press, Anchorage, USA, May 16-20, 2011 (acceptance: 112 of 571).